**网络爬虫——针对任意主题批量爬取PDF:** [**https://www.cnblogs.com/scikit-learn/p/7163721.html**](https://www.cnblogs.com/scikit-learn/p/7163721.html)

**任务需求**：要求通过Google针对任意关键字爬取大量PDF文档，如K-means，KNN，SVM等。

环境：Anaconda3——Windows7-64位——Python3.6——lantern——迅雷极速版——360浏览器

解决思路：现在有一个需要下载大量特定主题的PDF文档以为特定工作提供数据支撑的任务，初步是想利用Google强大的搜索功能进行特定主题的搜索，指定文件类型filetype：pdf，如果你要指定site的话，如.edu、.org等，可以在特定的机构或学校的网站上搜索文档。比如我要下载大量svm的pdf文档，我在Google搜索栏中输入svm filetype:pdf之后分析每一页的地址，发现地址之间存在类似之处，这就为写代码批量下载PDF文档提供了便利。这是一个很通用的通过Google下载大量特定主题特定文档的爬虫，有很多用途，读者可以自己去挖掘噢！比如博主就用这个简单的爬虫正在完成导师交代的任务，当然这只是其中很小一步，但是总得一步一步往前走嘛！

先贴代码，下面再对特定行进行解释

[复制代码](javascript:void(0);)

# -\*- coding: utf-8 -\*-

import urllib.request

import re#导入正则表达式包

import time

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

url\_part\_1="https://www.google.co.jp/search?q=svm+filetype:pdf&lr=&hl=zh-CN&as\_qdr=all&ei=VIdnWZ3SGIS18QXW3aAg&start="

url\_part\_2="&sa=N&biw=1745&bih=810"

Page\_number=28 #每一个特定主题搜索结果的页数，这个针对不同的主题结果会不一样

#这几行代码针对不同的主题要进行相应的修改，稍后再解释url\_part\_1和url\_part\_2的意思

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

timeout=20 #设置网站的访问超时时间为20秒

sleep\_download\_time=5 #爬一个网页休息5秒钟，不能访问的太频繁

url\_list\_total=[]#将所有pdf文档的链接地址放在这个列表当中

user\_agent = 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36'

#这个是我的360浏览器的user\_agent,按F12可查

for i in range(0,Page\_number):

url\_page=url\_part\_1+str(i\*10)+url\_part\_2#拼接成每个Google搜索结果的页面地址，每个页面包含10个链接，故乘以10

headers = { 'User-Agent' : user\_agent,'Referer':url\_page}

request= urllib.request.Request(url\_page,headers=headers)

try:#可能会出现异常[WinError 10054]

page = urllib.request.urlopen(request,timeout=timeout)

except urllib.error.URLError as e:

print(e.reason)

html= page.read().decode('utf-8')#必须要有.decode('utf-8')以用来转码，从byte转为utf-8

pattern = re.compile('<h3 class="r".\*?<a href="(.\*?)" onmousedown')#通过compile方法编译生成一个匹配模式pattern对象，然后利用这个对象来进行进一步的匹配。

url\_list=re.findall(pattern,html)#搜索html，以列表形式返回全部能匹配的子串这个表达式主要是为了获取本页面的PDF链接地址

for line in url\_list[:]:

url\_list\_total.append(line)

url\_list=[]

print(page.getcode())

print(i)

page.close()

html=[]

time.sleep(sleep\_download\_time)#隔几秒再运行

f=open('url\_list\_total.txt','w')

for url in url\_list\_total:

f.write(url)

f.write('\n')

f.close()

[复制代码](javascript:void(0);)

注释：

**7-8行：**通过Google搜索的结果地址是有规律的，如通过关键字svm filetype:pdf返回的结果的第一页地址如下

https://www.google.co.jp/search?q=svm+filetype:pdf&lr=&hl=zh-CN&as\_qdr=all&ei=VIdnWZ3SGIS18QXW3aAg&start=10&sa=N&biw=1745&bih=810

做如下拆分

"https://www.google.co.jp/search?q=svm+filetype:pdf&lr=&hl=zh-CN&as\_qdr=all&ei=VIdnWZ3SGIS18QXW3aAg&start="     +    10     +       "&sa=N&biw=1745&bih=810"

分成三部分，方便后面进行迭代。

**21行：**用来包装头部的数据：

——User-Agent：头部可以携带浏览器名和版本号、操作系统名和版本号、默认语言信息；

——Referer：用来防止盗链

——Connection：表示连接状态，记录Session状态

##以上就完成了所有pdf链接的下载

------------------------------------------------------------------------------------------------------分割线---------------------------------------------------------------------------------------------------------------

下面就进行pdf的主动下载，代码如下：

[复制代码](javascript:void(0);)

def getFile(url):

file\_name = url.split('/')[-1]

u = urllib.request.urlopen(url)

f = open(file\_name, 'wb')

block\_sz = 8192

while True:

buffer = u.read(block\_sz)

if not buffer:

break

f.write(buffer)

f.close()

print("Sucessful to svm" + " " + file\_name)

os.mkdir('svm')

os.chdir(os.path.join(os.getcwd(), 'svm'))

for url in url\_list\_total[:]:

getFile(url)

[复制代码](javascript:void(0);)

我在下载的时候并没有用以上的代码，而是直接通过迅雷极速版下载的，应该会快一些，当然了，用代码下载能省一些事儿！

欢迎批评指正！